

DELTA CON: A Principled Massive-Graph Similarity Function

Danai Koutra*
danai@cs.cmu.edu

Joshua T. Vogelstein†
jovo@stat.duke.edu

Christos Faloutsos*
christos@cs.cmu.edu

Abstract

How much did a network change since yesterday? How different is the wiring between Bob’s brain (a left-handed male) and Alice’s brain (a right-handed female)? Graph similarity with known node correspondence, i.e. the detection of changes in the connectivity of graphs, arises in numerous settings. In this work, we formally state the axioms and desired properties of the graph similarity functions, and evaluate when state-of-the-art methods fail to detect crucial connectivity changes in graphs. We propose DELTA CON, a principled, intuitive, and scalable algorithm that assesses the similarity between two graphs on the same nodes (e.g. employees of a company, customers of a mobile carrier). Experiments on various synthetic and real graphs showcase the advantages of our method over existing similarity measures. Finally, we employ DELTA CON to real applications: (a) we classify people to groups of high and low creativity based on their brain connectivity graphs, and (b) do temporal anomaly detection in the who-emails-whom Enron graph.

1 Introduction

Graphs arise naturally in numerous situations; social, traffic, collaboration and computer networks, images, protein-protein interaction networks, brain connectivity graphs and web graphs are only a few examples. A problem that comes up often in all those settings is the following: how much do two graphs or networks differ in terms of connectivity?

Graph similarity (or comparison) is a core task for sense-making: abnormal changes in the network traffic may indicate a computer attack; differences of big extent in a who-calls-whom graph may reveal a national celebration, or a telecommunication problem. Besides, network similarity can give insights into behavioral patterns: is the Facebook message graph similar to the Facebook wall-to-wall graph? Tracking changes in networks over time, spotting anomalies and detecting events is a research direction that has attracted much interest (e.g., [1], [2], [3]).

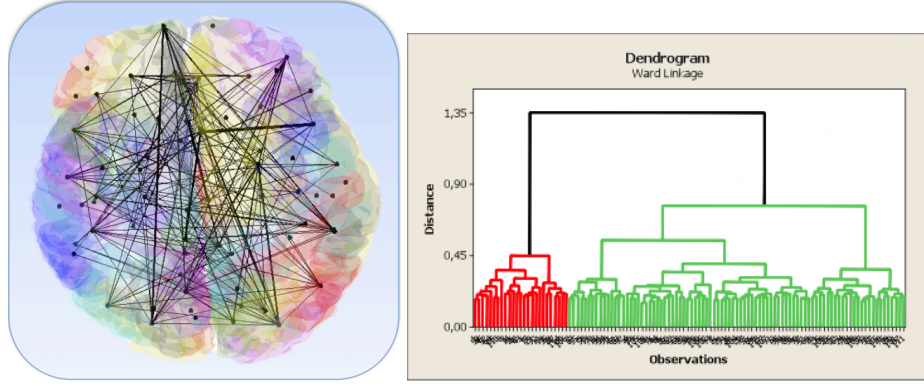
Long in the purview of researchers, graph similarity is a well-studied problem and several approaches have been proposed to solve variations of the problem. However, graph comparison still remains an open problem, while, with the passage of time, the list of requirements increases: the exponential growth of graphs, both in number and size, calls for methods that are not only accurate, but also scalable to graphs with billions of nodes.

In this paper, we address two main questions: How to compare two networks efficiently? How to evaluate their similarity score? Our main contributions are the following:

1. *Axioms/Properties*: we formalize the axioms and properties that a similarity measure must conform to.
2. *Algorithm*: we propose DELTA CON for measuring connectivity differences between two graphs, and show that it is: (a) *principled*, conforming to all the axioms presented in Section 2, (b) *intuitive*, giving similarity scores that agree with common sense and can be easily explained, and (c) *scalable*, able to handle large-scale graphs.

*Computer Science Department, Carnegie Mellon University.

†Department of Statistical Science, Duke University.



(a) Connectome: neural network of brain. (b) Dendrogram representing the hierarchical clustering of the DELTACON similarities between the 114 connectomes.

Figure 1: (a) Brain network (connectome). Different colors correspond to each of the 70 cortical regions, whose centers are depicted by vertices. Connections between the regions are shown by edges. DELTACON is used for clustering and classification. (b) The connectomes are nicely classified in two big clusters by hierarchical clustering. The classification is based on the pairwise DELTACON similarities between the 114 connectomes that we study. Elements in red correspond to high artistic score - thus, DELTACON shows that artistic brains seem to have different wiring than the rest.

3. *Experiments*: we report experiments on synthetic and real datasets, and compare DELTACON to six state-of-the-art methods that apply to our setting.
4. *Applications*: We use DELTACON for real-world applications, such as temporal anomaly detection and clustering/classification. In Fig. 1, DELTACON is used for clustering brain graphs corresponding to 114 individuals; the two big clusters which differ in terms of connectivity correspond to people with high and low creativity. More details are given in Sec. 5.

The paper is organized as follows: Section 2 presents the intuition behind our method, and the axioms and desired properties of a similarity measure; Sec. 3 has the proposed algorithms; experiments on synthetic and big real networks are in Sec. 4; Sec. 5 presents two real-world applications; the related work and the conclusions are in Sec. 6 and 7 respectively. Finally, Table 1 presents the major symbols we use in the paper and their definitions.

2 Proposed Method: Intuition

How can we find the similarity in connectivity between two graphs, or more formally how can we solve the following problem?

PROBLEM 1. DELTAConnectivity

Given: (a) two graphs, $G_1(\mathcal{V}, \mathcal{E}_1)$ and $G_2(\mathcal{V}, \mathcal{E}_2)$ with the same node set¹, \mathcal{V} , and different edge sets \mathcal{E}_1 and \mathcal{E}_2 , and (b) the node correspondence.

Find: a similarity score, $\text{sim}(G_1, G_2) \in [0, 1]$, between the input graphs. Similarity score of value 0 means totally different graphs, while 1 means identical graphs.

The obvious way to solve this problem is by measuring the overlap of their edges. Why does this often not work in practice? Consider the following example: according to the overlap method, the pairs of barbell graphs shown in Fig. 2 of p. 8, $(B10, mB10)$ and $(B10, mmB10)$, have the same similarity score. But, clearly, from the aspect of information flow, a missing edge from a clique ($mB10$) does not play as important role in the graph connectivity as the missing “bridge” in $mmB10$. So, could we instead measure the differences in the 1-step

¹If the graphs have different, but overlapping, node sets \mathcal{V}_1 and \mathcal{V}_2 , we assume that $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2$, and the extra nodes are treated as singletons.

Table 1: Symbols and Definitions. Bold capital letters: matrices, lowercase letters with arrows: vectors, plain font: scalars.

Symbol	Description
G	graph
\mathcal{V}, n	set of nodes, number of nodes
\mathcal{E}, m	set of edges, number of edges
$sim(G_1, G_2)$	similarity between graphs G_1 and G_2
$d(G_1, G_2)$	distance between graphs G_1 and G_2
I	$n \times n$ identity matrix
A	$n \times n$ adjacency matrix with elements a_{ij}
D	$n \times n$ diagonal degree matrix, $d_{ii} = \sum_j a_{ij}$
L	$= \mathbf{D} - \mathbf{A}$ laplacian matrix
S	$n \times n$ matrix of final scores with elements s_{ij}
S'	$n \times g$ reduced matrix of final scores
\vec{e}_i	$n \times 1$ unit vector with 1 in the i^{th} element
\vec{s}_{0k}	$n \times 1$ vector of seed scores for group k
\vec{s}_i	$n \times 1$ vector of final affinity scores to node i
g	number of groups (node partitions)
ϵ	$= 1/(1 + \max_i (d_{ii}))$ positive constant (< 1) encoding the influence between neighbors
DC₀, DC	DELTA CON ₀ , DELTA CON
VEO	Vertex/Edge Overlap
GED	Graph Edit Distance [4]
SS	Signature Similarity [5]
λ-D ADJ. / LAP N.L.	λ -distance on A / L / normalized L

away neighborhoods, 2-step away neighborhoods etc.? If yes, with what weight? It turns out (Intuition 1) that our method does exactly this in a principled way.

2.1 Fundamental Concept The first conceptual step of our proposed method is to compute the pairwise node affinities in the first graph, and compare them with the ones in the second graph. For notational compactness, we store them in a $n \times n$ similarity matrix² \mathbf{S} . The s_{ij} entry of the matrix indicates the influence node i has on node j . For example, in a who-knows-whom network, if node i is, say, republican and if we assume homophily (i.e., neighbors are similar), how likely is it that node j is also republican? Intuitively, node i has more influence/affinity to node j if there are many, short, heavily weighted paths from node i to j .

The second conceptual step is to measure the differences in the corresponding node affinity scores of the two graphs and report the result as their similarity score.

2.2 How to measure node affinity? Pagerank, personalized Random Walks with Restarts (RWR), lazy RWR, and the “electrical network analogy” technique are only a few of the methods that compute node affinities. We could have used Personalized RWR: $[\mathbf{I} - (1 - c)\mathbf{A}\mathbf{D}^{-1}]\vec{s}_i = c\vec{e}_i$, where c is the probability of restarting the random walk from the initial node, \vec{e}_i the starting (seed) indicator vector (all zeros except 1 at position i), and \vec{s}_i the unknown Personalized Pagerank column vector. Specifically, s_{ij} is the affinity of node j w.r.t. node i . For reasons that we explain next, we chose to use a more recent and principled method, the so-called Fast Belief Propagation (FABP), which is identical to Personalized RWR under specific conditions (see Theorem 2 in Appendix A.2 of [6]). We use a simplified form of it (see Appendix A.1 in [6]) given by:

$$(2.1) \quad [\mathbf{I} + \epsilon^2\mathbf{D} - \epsilon\mathbf{A}]\vec{s}_i = \vec{e}_i$$

where $\vec{s}_i = [s_{i1}, \dots, s_{in}]^T$ is the column vector of final similarity/influence scores starting from the i^{th} node, ϵ is a small constant capturing the influence between neighboring nodes, \mathbf{I} is the identity matrix, \mathbf{A} is the adjacency matrix and \mathbf{D} is the diagonal matrix with the degree of node i as the d_{ii} entry.

An equivalent, more compact notation, is to use a matrix form, and to stack all the \vec{s}_i vectors ($i = 1, \dots, n$) into the $n \times n$ matrix \mathbf{S} . We can easily prove that

$$(2.2) \quad \boxed{\mathbf{S} = [s_{ij}] = [\mathbf{I} + \epsilon^2\mathbf{D} - \epsilon\mathbf{A}]^{-1}}.$$

2.3 Why use Belief Propagation? The reasons we choose BP and its fast approximation with Eq. (2.2) are: (a) it is based on sound theoretical background (maximum likelihood estimation on marginals), (b) it is fast (linear on the number of edges), and (c) it agrees with intuition, taking into account not only direct neighbors, but also 2-, 3- and k -step-away neighbors, with decreasing weight. We elaborate on the last reason, next:

INTUITION 1. [Attenuating Neighboring Influence]

By temporarily ignoring the term $\epsilon^2\mathbf{D}$ in (2.2), we can expand the matrix inversion and approximate the $n \times n$ matrix of pairwise affinities, \mathbf{S} , as

$$\mathbf{S} \approx [\mathbf{I} - \epsilon\mathbf{A}]^{-1} \approx \mathbf{I} + \epsilon\mathbf{A} + \epsilon^2\mathbf{A}^2 + \dots$$

As we said, our method captures the differences in the 1-step, 2-step, 3-step etc. neighborhoods in a weighted way; differences in long paths have smaller effect on the computation of the similarity measure than differences in short paths. Recall that $\epsilon < 1$, and that \mathbf{A}^k has information about the k -step paths. Notice that this is just the intuition behind our method; we do not use this simplified formula to find matrix \mathbf{S} .

²In reality, we don’t measure all the affinities (see Section 3.2 for an efficient approximation).

2.4 Which properties should a similarity measure satisfy? Let $G_1(\mathcal{V}, \mathcal{E}_1)$ and $G_2(\mathcal{V}, \mathcal{E}_2)$ be two graphs, and $\text{sim}(G_1, G_2) \in [0, 1]$ denote their similarity score. Then, we want the similarity measure to obey the following axioms:

- A1. *Identity property*: $\text{sim}(G_1, G_1) = 1$
- A2. *Symmetric property*: $\text{sim}(G_1, G_2) = \text{sim}(G_2, G_1)$
- A3. *Zero property*: $\text{sim}(G_1, G_2) \rightarrow 0$ for $n \rightarrow \infty$, where G_1 is the clique graph (K_n), and G_2 is the empty graph (i.e., the edge sets are complementary).

Moreover, the measure must be:

(a) **intuitive**. It should satisfy the following desired properties:

- P1. [*Edge Importance*] Changes that create disconnected components should be penalized more than changes that maintain the connectivity properties of the graphs.
- P2. [*Weight Awareness*] In weighted graphs, the bigger the weight of the removed edge is, the greater the impact on the similarity measure should be.
- P3. [*Edge-“Submodularity”*] A specific change is more important in a graph with few edges than in a much denser, but equally sized graph.
- P4. [*Focus Awareness*] Random changes in graphs are less important than targeted changes of the same extent.

(b) **scalable**. The huge size of the generated graphs, as well as their abundance require a similarity measure that is computed fast and handles graphs with billions of nodes.

3 Proposed Method: Details

Now that we have described the high level ideas behind our method, we move on to the details.

3.1 Algorithm Description Let the graphs we compare be $G_1(\mathcal{V}, \mathcal{E}_1)$ and $G_2(\mathcal{V}, \mathcal{E}_2)$. If the graphs have different node sets, say \mathcal{V}_1 and \mathcal{V}_2 , we assume that $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2$, where some nodes are disconnected.

As mentioned before, the main idea behind our proposed similarity algorithm is to compare the node affinities in the given graphs. The steps of our similarity method are:

Step 1. By eq. (2.2), we compute for each graph the $n \times n$ matrix of pairwise node affinity scores (\mathbf{S}_1 and \mathbf{S}_2 for graphs G_1 and G_2 respectively).

Step 2. Among the various distance and similarity measures (e.g., Euclidean distance (ED), cosine similarity, correlation) found in the literature, we use the root euclidean distance (ROOTED, a.k.a. Matusita distance)

$$(3.3) \quad d = \text{ROOTED}(\mathbf{S}_1, \mathbf{S}_2) = \sqrt{\sum_{i=1}^n \sum_{j=1}^n (\sqrt{s_{1,ij}} - \sqrt{s_{2,ij}})^2}.$$

We use the ROOTED distance for the following reasons:

1. it is very similar to the Euclidean distance (ED), the only difference being the square root of the pairwise similarities (s_{ij}),
2. it usually gives better results, because it “boosts” the node affinities³ and, therefore, detects even small changes in the graphs (other distance measures, including ED, suffer from high similarity scores no matter how much the graphs differ), and
3. satisfies the desired properties P1-P4. As discussed in the Appendix A.5 of [6], at least P1 is not satisfied by the ED.

³The node affinities are in $[0, 1]$, so the square root makes them bigger.

Step 3. For interpretability, we convert the distance (d) to similarity measure (sim) via the formula $sim = \frac{1}{1+d}$. The result is bounded to the interval $[0,1]$, as opposed to being unbounded $[0,\infty)$. Notice that the distance-to-similarity transformation does *not* change the ranking of results in a nearest-neighbor query.

The straightforward algorithm, DELTA CON_0 (Algorithm 1), is to compute all the n^2 affinity scores of matrix S by simply using equation (2.2). We can do the inversion using the Power Method or any other efficient method.

Algorithm 1 DELTA CON_0

```

INPUT: edge files of  $G_1(\mathcal{V}, \mathcal{E}_1)$  and  $G_2(\mathcal{V}, \mathcal{E}_2)$ 
//  $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2$ , if  $\mathcal{V}_1$  and  $\mathcal{V}_2$  are the graphs' node sets
 $S_1 = [I + \epsilon^2 D_1 - \epsilon A_1]^{-1}$                                 //  $s_{1,ij}$ : affinity/influence of
 $S_2 = [I + \epsilon^2 D_2 - \epsilon A_2]^{-1}$                                 // node  $i$  to node  $j$  in  $G_1$ 
 $d(G_1, G_2) = \text{ROOTED}(S_1, S_2)$ 
return  $sim(G_1, G_2) = \frac{1}{1+d}$ 

```

3.2 Scalability Analysis DELTA CON_0 satisfies all the properties in Section 2, but it is quadratic (n^2 affinity scores s_{ij} - using power method for the inversion of sparse matrix) and thus not scalable. We present a faster, linear algorithm, DELTA CON (Algorithm 2), which approximates DELTA CON_0 and differs in the first step. We still want each node to become a seed exactly once in order to find the affinities of the rest of the nodes to it; but, here, we have multiple seeds at once, instead of having one seed at a time. The idea is to randomly divide our node-set into g groups, and compute the affinity score of each node i to group k , thus requiring only $n \times g$ scores, which are stored in the $n \times g$ matrix S' ($g \ll n$). Intuitively, instead of using the $n \times n$ affinity matrix S , we add up the scores of the columns that correspond to the nodes of a group, and obtain the $n \times g$ matrix S' ($g \ll n$). The score s'_{ik} is the affinity of node i to the k^{th} group of nodes ($k = 1, \dots, g$).

LEMMA 3.1. *The time complexity of computing the reduced affinity matrix, S' , is linear on the number of edges.*

Proof. We can compute the $n \times g$ “skinny” matrix S' quickly, by solving $[I + \epsilon^2 D - \epsilon A]S' = [\vec{s}_{01} \dots \vec{s}_{0g}]$, where $\vec{s}_{0k} = \sum_{i \in group_k} \vec{e}_i$ is the membership $n \times 1$ vector for group k (all 0's, except 1's for members of the group). \square

Thus, we compute g final scores per node, which denote its affinity to every *group* of seeds, instead of every seed node that we had in eq. (2.2). With careful implementation, DELTA CON is linear on the number of number of edges and groups g . As we show in section 4.2, it takes ~ 160 sec, on commodity hardware, for a 1.6-million-node graph.

Once we have the reduced affinity matrices S'_1 and S'_2 of the two graphs, we use the ROOTED, to find the similarity between the $n \times g$ matrices of final scores, where $g \ll n$.

LEMMA 3.2. *The time complexity of DELTA CON , when applied in parallel to the input graphs, is linear on the number of edges in the graphs, i.e. $O(g \cdot \max\{m_1, m_2\})$.*

Proof. Based on lemma 3.1. See Appendix A.3 in [6]. \square

THEOREM 1. *DELTA CON 's similarity score between any two graphs G_1, G_2 upper bounds the actual DELTA CON_0 's similarity score, i.e. $sim_{DC-0}(G_1, G_2) \leq sim_{DC}(G_1, G_2)$.*

Proof. Intuitively, grouping nodes blurs the influence information and makes the nodes seem more similar than originally. For more details, see Appendix A.3 of [6]. \square

In the following section we show that DELTA CON (which includes DELTA CON_0 as a special case for $g = n$) satisfies the axioms and properties, while in the Appendix (A.4 and A.5 in [6]) we provide the proofs.

Algorithm 2 DELTACON

INPUT: edge files of $G_1(\mathcal{V}, \mathcal{E}_1)$ and $G_2(\mathcal{V}, \mathcal{E}_2)$ and
 g (groups: # of node partitions)

$\{\mathcal{V}_j\}_{j=1}^g = \text{random_partition}(\mathcal{V}, g)$

// g groups

// estimate affinity vector of nodes $i = 1, \dots, n$ to group k

for $k = 1 \rightarrow g$ **do**

$\vec{s}_{0k} = \sum_{i \in \mathcal{V}_k} \vec{e}_i$

solve $[\mathbf{I} + \epsilon^2 \mathbf{D}_1 - \epsilon \mathbf{A}_1] \vec{s}'_{1k} = \vec{s}_{0k}$

solve $[\mathbf{I} + \epsilon^2 \mathbf{D}_2 - \epsilon \mathbf{A}_2] \vec{s}'_{2k} = \vec{s}_{0k}$

end for

$\mathbf{S}'_1 = [\vec{s}'_{11} \ \vec{s}'_{12} \ \dots \ \vec{s}'_{1g}]$; $\mathbf{S}'_2 = [\vec{s}'_{21} \ \vec{s}'_{22} \ \dots \ \vec{s}'_{2g}]$

// compare affinity matrices \mathbf{S}'_1 and \mathbf{S}'_2

$d(G_1, G_2) = \text{ROOTED}(\mathbf{S}'_1, \mathbf{S}'_2)$

return $\text{sim}(G_1, G_2) = \frac{1}{1+d}$

Table 2: Real and Synthetic Datasets

Name	Nodes	Edges	Description
Brain Graphs	70	800-1,208	connectome
Enron Email [7]	36,692	367,662	who-emails-whom
Epinions [8]	131,828	841,372	who-trusts-whom
Email EU [9]	265,214	420,045	who-sent-to-whom
Web Google [10]	875,714	5,105,039	site-to-site
AS skitter [9]	1,696,415	11,095,298	p2p links
Kronecker 1	6,561	65,536	synthetic
Kronecker 2	19,683	262,144	synthetic
Kronecker 3	59,049	1,048,576	synthetic
Kronecker 4	177,147	4,194,304	synthetic
Kronecker 5	531,441	16,777,216	synthetic
Kronecker 6	1,594,323	67,108,864	synthetic

4 Experiments

We conduct several experiments on synthetic and real data (undirected, unweighted graphs, unless stated otherwise - see Table 2) to answer the following questions:

Q1. Does DELTACON agree with our intuition and satisfy the axioms/properties? Where do other methods fail?

Q2. Is DELTACON scalable?

The implementation is in Matlab and we ran the experiments on AMD Opteron Processor 854 @3GHz, RAM 32GB.

4.1 Intuitiveness of DELTACON. To answer Q1, for the first 3 properties (P1-P3), we conduct experiments on small graphs of 5 to 100 nodes and classic topologies (cliques, stars, circles, paths, barbell and wheel-barbell graphs, and “lollipops” shown in Fig. 2), since people can argue about their similarities. For the name conventions see Table 3. For our method we used 5 groups (g), but the results are similar for other choices of the parameter. In addition to the synthetic graphs, for the last property (P4), we use real networks with up to 11 million edges (Table 2).

We compare our method, DELTACON, to the 6 best state-of-the-art similarity measures that apply to our

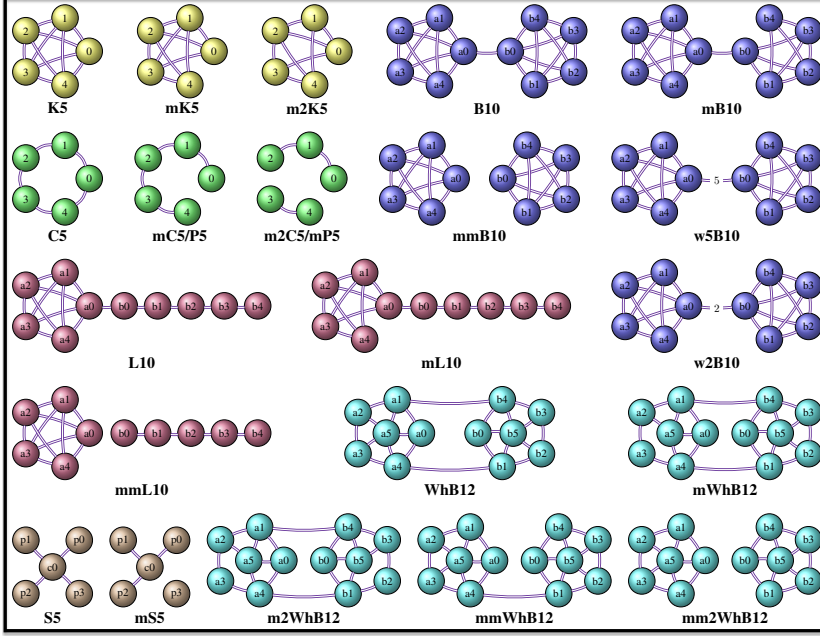


Figure 2: Small synthetic graphs – *K*: clique, *C*: cycle, *P*: path, *S*: star, *B*: barbell, *L*: lollipop, *WhB*: wheel-barbell

Symbol	Meaning
K_n	clique of size n
P_n	path of size n
C_n	cycle of size n
S_n	star of size n
L_n	lollipop of size n
B_n	barbell of size n
WhB_n	wheel barbell of size n
m_X	missing X edges
mm_X	missing X “bridge” edges
w	weight of “bridge” edge

Table 3: Name Conventions for small synthetic graphs. Missing number after the prefix implied $X = 1$.

setting:

1. Vertex/Edge Overlap (VEO) [5]: For two graphs $G_1(\mathcal{V}_1, \mathcal{E}_1)$ and $G_2(\mathcal{V}_2, \mathcal{E}_2)$:

$$sim_{VEO}(G_1, G_2) = 2 \frac{|\mathcal{E}_1 \cap \mathcal{E}_2| + |\mathcal{V}_1 \cap \mathcal{V}_2|}{|\mathcal{E}_1| + |\mathcal{E}_2| + |\mathcal{V}_1| + |\mathcal{V}_2|}.$$

2. Graph Edit Distance (GED) [4]: GED has quadratic complexity in general, so they [4] consider the case where only insertions and deletions are allowed.

$$sim_{GED}(G_1, G_2) = |\mathcal{V}_1| + |\mathcal{V}_2| - 2|\mathcal{V}_1 \cap \mathcal{V}_2| + |\mathcal{E}_1| + |\mathcal{E}_2| - 2|\mathcal{E}_1 \cap \mathcal{E}_2|.$$

For $\mathcal{V}_1 = \mathcal{V}_2$ and unweighted graphs, sim_{GED} is equivalent to hamming distance($\mathbf{A}_1, \mathbf{A}_2$) = $sum(\mathbf{A}_1 XOR \mathbf{A}_2)$.

3. Signature Similarity (SS) [5]: This is the best performing similarity measure studied in [5]. It is based on the SimHash algorithm (random projection based method).
4. The last 3 methods are variations of the well-studied spectral method “ λ -distance” ([4], [11], [12]). Let $\{\lambda_{1i}\}_{i=1}^{|\mathcal{V}_1|}$ and $\{\lambda_{2i}\}_{i=1}^{|\mathcal{V}_2|}$ be the eigenvalues of the matrices that represent G_1 and G_2 . Then, λ -distance is given by

$$d_\lambda(G_1, G_2) = \sqrt{\sum_{i=1}^k (\lambda_{1i} - \lambda_{2i})^2},$$

where k is $\max(|\mathcal{V}_1|, |\mathcal{V}_2|)$ (padding is required for the smallest vector of eigenvalues). The variations of the method are based on three different matrix representations of the graphs: adjacency (λ -d Adj.), laplacian (λ -d Lap.) and normalized laplacian matrix (λ -d N.L.).

Table 4: “Edge Importance” (P1). Highlighted entries violate Table 5: “Weight Awareness” (P2). Highlighted entries violate P1.

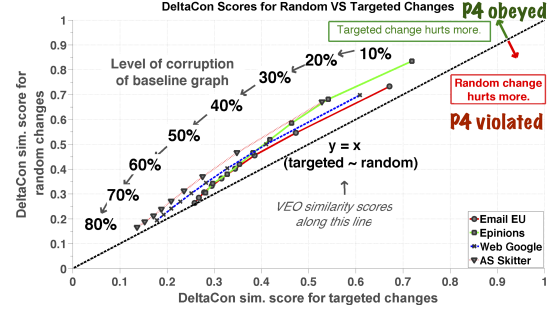
GRAPHS				DC ₀	DC	VEO	SS	GED (XOR)	λ -D Adj.	λ -D LAP.	λ -D N.L.
A	B	C	D	$\Delta s = \text{sim}(A, B) - \text{sim}(A, C)$				$\Delta d = d(A, C) - d(A, B)$			
B10	mB10	mmB10		0.07	0.04	0	-10^{-5}	0	0.21	-0.27	2.14
L10	mL10	mmL10		0.04	0.02	0	-10^{-5}	0	-0.30	-0.43	-8.23
WhB10	mWhB10	mmWhB10		0.03	0.01	0	-10^{-5}	0	0.22	0.18	-0.41
WhB10	m2WhB10	mm2WhB10		0.07	0.04	0	-10^{-5}	0	0.59	0.41	0.87

Table 6: “Edge-Submodularity” (P3). Highlighted entries violate P3.

GRAPHS				DC ₀	DC	VEO	SS	GED (XOR)	λ -D Adj.	λ -D LAP.	λ -D N.L.
A	B	C	D	$\Delta s = \text{sim}(A, B) - \text{sim}(C, D)$				$\Delta d = d(C, D) - d(A, B)$			
K5	mK5	C5	mC5	0.03	0.03	0.02	10^{-5}	0	-0.24	-0.59	-7.77
C5	mC5	P5	mP5	0.03	0.01	0.01	-10^{-5}	0	-0.55	-0.39	-0.20
P5	mP5	S5	mS5	0.003	0.001	0	-10^{-5}	0	-0.07	0.39	3.64
K ₁₀₀	mK ₁₀₀	C ₁₀₀	mC ₁₀₀	0.03	0.02	0.002	10^{-5}	0	-1.16	-1.69	-3.11
C ₁₀₀	mC ₁₀₀	P ₁₀₀	mP ₁₀₀	10^{-4}	0.01	10^{-5}	-10^{-5}	0	-0.08	-0.06	-0.08
P ₁₀₀	mP ₁₀₀	S ₁₀₀	mS ₁₀₀	0.05	0.03	0	0	0	-0.08	1.16	196
K ₁₀₀	m10K ₁₀₀	C ₁₀₀	m10C ₁₀₀	0.10	0.08	0.02	10^{-5}	0	-3.48	-4.52	-1089
C ₁₀₀	m10C ₁₀₀	P ₁₀₀	m10P ₁₀₀	0.001	0.001	10^{-5}	0	0	-0.03	0.01	0.31
P ₁₀₀	m10P ₁₀₀	S ₁₀₀	m10S ₁₀₀	0.13	0.07	0	-10^{-5}	0	-0.18	8.22	1873

GRAPHS				DC ₀	DC	VEO	SS	GED (XOR)	λ -D Adj.	λ -D LAP.	λ -D N.L.
A	B	C	D	$\Delta s = \text{sim}(A, B) - \text{sim}(C, D)$				$\Delta d = d(C, D) - d(A, B)$			
B10	mB10	B10	w5B10	0.09	0.08	-0.02	10^{-5}	-1	3.67	5.61	84.44
mmB10	B10	mmB10	w5B10	0.10	0.10	0	10^{-4}	0	4.57	7.60	95.61
B10	mB10	w5B10	w2B10	0.06	0.06	-0.02	10^{-5}	-1	2.55	3.77	66.71
w5B10	w2B10	w5B10	mmB10	0.10	0.07	0.02	10^{-5}	1	2.23	3.55	31.04
w5B10	w2B10	w5B10	B10	0.03	0.02	0	10^{-5}	0	1.12	1.84	17.73

Figure 3: “Focus-Awareness” (P4).



Tables 4-6/Figure 3: DELTA_{CON}₀ and DELTA_{CON} (in bold) obey all the required properties (P1-P4). Tables 4-6: Each row of the tables corresponds to a comparison between the similarities (or distances) of two pairs of graphs; pairs (A,B) and (A,C) for (P1); and pairs (A,B) and (C,D) for (P2) and (P3): Non-positive values of $\Delta s = \text{sim}(A, B) - \text{sim}(C, D)$ and $\Delta d = d(C, D) - d(A, B)$ - depending on whether the corresponding method computes similarity or distance - are highlighted and mean violation of the property of interest. Figure 3: Targeted changes hurt more than random. Plot of DELTA_{CON} similarity scores for random changes (y axis) vs. DELTA_{CON} similarity scores for targeted changes (x axis) for 4 real-world networks. For each graph we create 8 “corrupted” versions with 10% to 80% fewer edges than the initial graphs. Notice that all the points are above the diagonal.

The results for the first 3 properties are presented in the form of tables 4-6. For property P1 we compare the graphs (A,B) and (A,C) and report the difference between the pairwise similarities/distances of our proposed methods and the 6 state-of-the-art methods. We have arranged the pairs of graphs in such way that (A,B) are more similar than (A,C). Therefore, table entries that are non-positive mean that the corresponding method does not satisfy the property. Similarly, for properties P2 and P3, we compare the graphs (A,B) and (C,D) and report the difference in their pairwise similarity/distance scores.

P1. Edge Importance : “Edges whose removal creates disconnected components are more important than other edges whose absence does not affect the graph connectivity. The more important an edge is, the more it should affect the similarity or distance measure.”

For this experiment we use the barbell, “wheel barbell” and “lollipop” graphs, since it is easy to argue about the importance of the individual edges. The idea is that edges in a highly connected component (e.g. clique, wheel) are not very important from the information flow viewpoint, while edges that connect (almost uniquely) dense components play a significant role in the connectivity of the graph and the information flow. The importance of the “bridge” edge depends on the size of the components that it connects; the bigger the components the more important is the role of the edge.

OBSERVATION 1. Only DELTA_{CON} succeeds in distinguishing the importance of the edges (P1) w.r.t. connectivity, while all the other methods fail at least once (Table 4).

P2. Weight Awareness : “The absence of an edge of big weight is more important than the absence of a smaller weighted edge; this should be reflected in the similarity measure.”

The weight of an edge defines the strength of the connection between two nodes, and, in this sense, can be viewed as a feature that relates to the importance of the edge in the graph. For this property, we study the weighted versions of the barbell graph, where we assume that all the edges except the “bridge” have unit weight.

OBSERVATION 2. *All the methods are weight-aware (P2), except VEO and GED which compute just the overlap in edges and vertices between the graphs (Table 5).*

P3. “Edge-Submodularity” : “Let $A(\mathcal{V}, \mathcal{E}_1)$ and $B(\mathcal{V}, \mathcal{E}_2)$ be two graphs with the same node set, and $|\mathcal{E}_1| > |\mathcal{E}_2|$ edges. Also, assume that $m_x A(\mathcal{V}, \mathcal{E}_1)$ and $m_x B(\mathcal{V}, \mathcal{E}_2)$ are the respective derived graphs after removing x edges. We expect that $\text{sim}(A, m_x A) > \text{sim}(B, m_x B)$, since the fewer the edges in a constant-sized graph, the more “important” they are.”

The results for different graph topologies and 1 or 10 removed edges (prefixes ‘m’ and ‘m10’ respectively) are given compactly in Table 6. Recall that non-positive values denote violation of the “edge-submodularity” property.

OBSERVATION 3. *Only DELTACON complies to the “edge-submodularity” property (P3) in all cases examined.*

P4. Focus Awareness : At this point, all the competing methods have failed in satisfying at least one of the desired properties. To test whether DELTACON is able to distinguish the extent of a change in a graph, we analyze real datasets with up to 11 million edges (Table 2) for two different types of changes. For each graph we create corrupted instances by removing: (i) edges from the original graph randomly, and (ii) the same number of edges in a targeted way (we randomly choose nodes and remove all their edges, until we have removed the appropriate fraction of edges).

In Fig. 3, for each of the 4 real networks -Email EU, Enron, Google web and AS Skitter-, we give the pair ($\text{sim_DELTA_CON random}$, $\text{sim_DELTA_CON targeted}$) for each of the different levels of corruption (10%, 20%, ..., 80%). That is, for each corruption level and network, there is a point with coordinates the similarity score between the original graph and the corrupted graph when the edge removal is random, and the score when the edge removal is targeted. The line $y = x$ corresponds to equal similarity scores for both ways of removing edges.

OBSERVATION 4. • “Targeted changes hurt more.” DELTACON *is focus-aware (P4). Removal of edges in a targeted way leads to smaller similarity of the derived graph to the original one than removal of the same number of edges in a random way.*

- “More changes: random \approx targeted.” *As the corruption level increases, the similarity score for random changes tends to the similarity score for targeted changes (in Fig. 3, all lines converge to the $y = x$ line for greater level of corruption).*

This is expected as the random and targeted edge removal tend to be equivalent when a significant fraction of edges is deleted.

General Remarks. All in all, the baseline methods have several non-desirable properties. The spectral methods, as well as SS fail to comply to the “edge importance” (P1) and “edge submodularity” (P3) properties. Moreover, λ -distance has high computational cost when the whole graph spectrum is computed, cannot distinguish the differences between co-spectral graphs, and sometimes small changes lead to big differences in the graph spectra. As far as VEO and GED are concerned, they are oblivious on significant structural properties of the graphs; thus, despite their straightforwardness and fast computation, they fail to discern various changes in the graphs. On the other hand, DELTACON gives tangible similarity scores and conforms to all the desired properties.

4.2 Scalability of DELTACON. In Section 2 we demonstrated that DELTACON is linear on the number of edges, and here we show that this also holds in practice. We ran DELTACON on Kronecker graphs (Table 2), which are known [13] to share many properties with real graphs.

OBSERVATION 5. *As shown in Fig. 4, DELTACON scales linearly with the number of edges in the graph.*

Notice that the algorithm can be trivially parallelized by finding the node affinity scores of the two graphs in parallel instead of sequential. Moreover, for each graph the computation of the similarity scores of the nodes to each of the g groups can be parallelized. However, the runtime of our experiments refer to the sequential implementation.

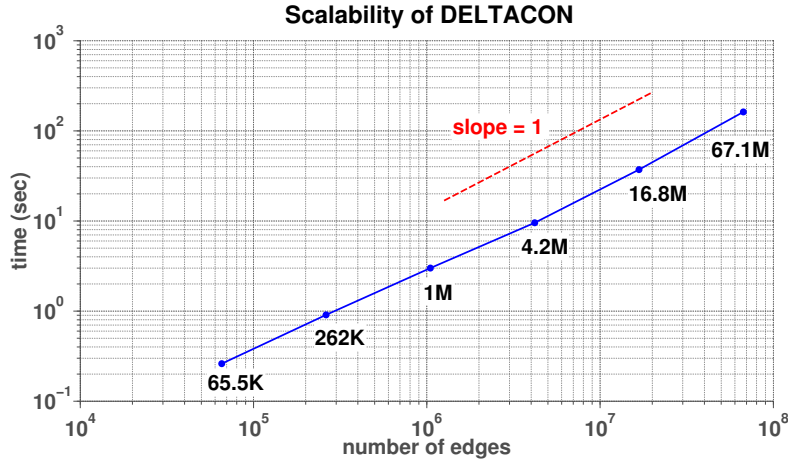


Figure 4: DELTACON is linear on the number of edges (time in sec. vs. number of edges). The exact number of edges is annotated.

5 DELTACON at Work

In this section we present two applications of graph similarity measures; we use DELTACON and report our findings.

5.1 Enron. First, we analyze the time-evolving ENRON graph. Figure 5 depicts the similarity scores between consecutive daily who-emailed-whom graphs. By applying Quality Control with Individual Moving Range, we obtain the lower and upper limits of the in-control similarity scores. These limits correspond to median $\pm 3\sigma$ (The median is used instead of the mean, since appropriate hypothesis tests demonstrate that the data does not follow the normal distribution. Moving range mean is used to estimate σ). Using this method we were able to define the threshold (lower control limit) below which the corresponding days are anomalous, i.e. they differ “too much” from the previous and following days. Note that all the anomalous days relate to crucial events in the company’s history during 2001 (points marked with red boxes in Fig. 5): (2) 8/21, Lay emails all employees stating he wants “to restore investor confidence in Enron.”; (3) 9/26, Lay tells employees that the accounting practices are “legal and totally appropriate”, and that the stock is “an incredible bargain.”; (4) 10/5, Just before Arthur Andersen hired Davis Polk & Wardwell law firm to prepare a defense for the company; (5) 10/24-25, Jeff McMahon takes over as CFO. Email to all employees states that all the pertinent documents should be preserved; (6) 11/8, Enron announces it overstated profits by 586 million dollars over 5 years.

Although high similarities between consecutive days do not consist anomalies, we found that mostly weekends expose high similarities. For instance, the first two points of 100% similarity correspond to the weekend before Christmas in 2000 and a weekend in July, when only two employees sent emails to each other. It is noticeable that after February 2002 many consecutive days are very similar; this happens because, after the

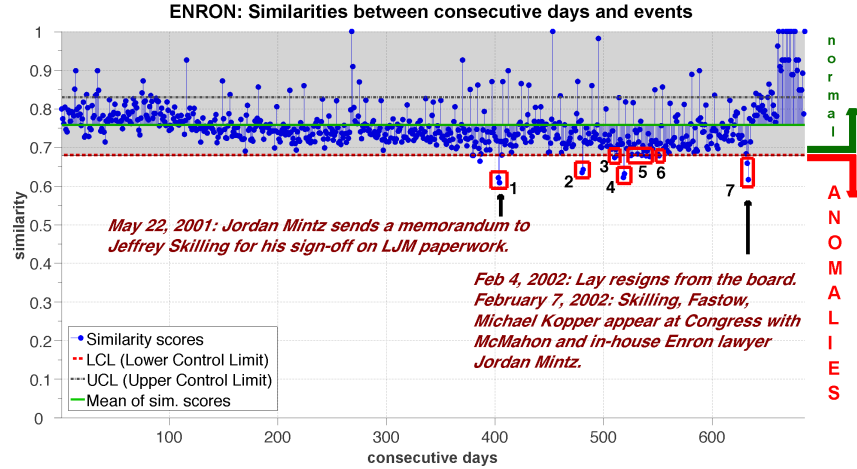


Figure 5: Graph Anomaly Detection with DELTACON. The marked days correspond to anomalies and coincide with major events in the history of Enron. The blue points are similarity scores between consecutive instances of the daily email activity between the employees, and the marked days are 3σ units away from the median similarity score.

collapse of Enron, the email exchange activity was rather low and often between certain employees.

5.2 Brain Connectivity Graph Clustering. We also use DELTACON for clustering and classification. For this purpose we study *connectomes* -brain graphs-, which are obtained by Multimodal Magnetic Resonance Imaging [14].

In total we study the connectomes of 114 people; each consists of 70 cortical regions (nodes), and connections (weighted edges) between them. We ignore the strength of connections and derive an undirected, unweighted brain graph per person. In addition to the connectomes, we have attributes for each person (e.g., age, gender, IQ).

We first get the DELTACON pairwise similarities between the brain graphs, and then perform hierarchical clustering using Ward’s method (Fig. 1(b)). As shown in the figure, there are two clearly separable groups of brain graphs. Applying t-test on the available attributes for the two groups created by the clusters, we found that the latter differ significantly (p-value=0.0057) in the Composite Creativity Index (CCI), which is related to the person’s performance on a series of creativity tasks. Moreover, the two groups correspond to low and high openness index (p-value=0.0558), one of the “Big Five Factors”; that is, the brain connectivity is different in people that are inventive and people that are consistent. Exploiting analysis of variance (ANOVA: generalization of t-test when more than 2 groups are analyzed), we tested whether the various clusters that we obtain from hierarchical clustering reflect the structural differences in the brain graphs. However, in the dataset we studied there is no sufficient statistical evidence that age, gender, IQ etc. are related to the brain connectivity.

6 Related Work

Graph Similarity. The problems are divided in two main categories: (1) *With Known Node Correspondence*. Papadimitriou et al. [5] propose 5 similarity measures for directed web graphs. Among them the best is the Signature Similarity (SS), which is based on the SimHash algorithm, while the Vertex/Edge Overlap similarity (VEO) performs very well. Bunke [4] presents techniques used to track sudden changes in communications networks for performance monitoring. The best approaches are the Graph Edit Distance and Maximum Common Subgraph. Both are NP-complete, but the former approach can be simplified given the application and it becomes linear on the number of nodes and edges in the graphs. (2) *With Unknown Node Correspondence*. Two approaches can be used: (a) feature extraction and similarity computation, (b) graph matching and application of techniques from the first category [15], (c) graph kernels [16]. The research directions in this category include: λ -distance ([4],[11],[12]), a spectral method that has been studied thoroughly; algebraic connectivity [17]; an SVM-based

approach on global feature vectors [18]; social networks similarity [19]; computing edge curvatures under heat kernel embedding [20]; comparison of the number of spanning trees [21]; fast random walk graph kernel [22].

Both research directions are important, but apply in different settings; if the node correspondence is available, the algorithms that make use of it can perform only better than methods that omit it. Here we tackle the former problem.

Node affinity algorithms. There are numerous node affinity algorithms; Pagerank [23], Personalized Random Walks with Restarts [24], the electric network analogy [25], SimRank [26], and Belief Propagation [27] are only some examples of the most successful techniques. Here we focus on the latter method, and specifically a fast variation [28] which is also intuitive. All the techniques have been used successfully in many tasks, such as ranking, classification, malware and fraud detection ([29],[30]), and recommendation systems [31].

7 Conclusions

In this work, we tackle the problem of graph similarity when the node correspondence is known (e.g., similarity in time-evolving phone networks). Our contributions are:

- *Axioms/Properties:* we formalize the problem of graph similarity by providing axioms, and desired properties.
- *Algorithm:* We propose DELTACON, an algorithm that is (a) *principled* (axioms $A1-A3$, in Sec. 2), (b) *intuitive* (properties $P1-P4$, in Sec. 4), and (c) *scalable*, needing on commodity hardware ~ 160 seconds for a graph with over 67 million edges.
- *Experiments:* We evaluate the intuitiveness of DELTACON, and compare it to 6 state-of-the-art measures.
- *Applications:* We use DELTACON for temporal anomaly detection (ENRON), and clustering & classification (brain graphs).

Future work includes parallelizing our algorithm, as well as trying to partition the graphs in a more informative way (e.g., using elimination tree) than random.

8 Acknowledgements

The authors would like to thank Aaditya Ramdas, Aarti Singh, Elijah Mayfield, Gary Miller, and Jilles Vreeken for their helpful comments and suggestions.

Funding was provided by the U.S. Army Research Office (ARO) and Defense Advanced Research Projects Agency (DARPA) under Contract Number W911NF-11-C-0088. Research was also sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-09-2-0053. It was also partially supported by an IBM Faculty Award. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

- [1] R. S. Caceres, T. Y. Berger-Wolf, and R. Grossman, “Temporal scale of processes in dynamic networks,” in *ICDM Workshop*, 2011, pp. 925–932.
- [2] C. C. Noble and D. J. Cook, “Graph-based anomaly detection,” in *Proceedings of the 9th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, Washington, DC, 2003, pp. 631–636.
- [3] Y. Wang, S. Parthasarathy, and S. Tatikonda, “Locality sensitive outlier detection: A ranking driven approach,” in *Proceedings of the 27th International Conference on Data Engineering (ICDE)*, Hannover, Germany, 2011, pp. 410–421.
- [4] H. Bunke, P. J. Dickinson, M. Kraetzl, and W. D. Wallis, *A Graph-Theoretic Approach to Enterprise Network Dynamics (PCS)*. Birkhauser, 2006.
- [5] P. Papadimitriou, A. Dasdan, and H. Garcia-Molina, “Web graph similarity for anomaly detection,” *Journal of Internet Services and Applications*, vol. 1, no. 1, p. 1167, 2008.
- [6] [Online]. Available: http://www.cs.cmu.edu/~dkoutra/papers/DeltaCon_KoutraVF_withAppendix.pdf

- [7] B. Klimt and Y. Yang, “Introducing the enron corpus.” 2004.
- [8] R. Guha, R. Kumar, P. Raghavan, and A. Tomkins, “Propagation of trust and distrust.” ACM, 2004, pp. 403–412.
- [9] J. Leskovec, J. Kleinberg, and C. Faloutsos, “Graph evolution: Densification and shrinking diameters,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 1, March 2007.
- [10] SNAP. [Online]. Available: <http://snap.stanford.edu/data/index.html>
- [11] M. Peabody, “Finding groups of graphs in databases,” Master’s thesis, Drexel University, 2003.
- [12] R. C. Wilson and P. Zhu, “A study of graph spectra for comparing graphs and trees,” *Journal of Pattern Recognition*, vol. 41, no. 9, pp. 2833–2841, 2008.
- [13] J. Leskovec, D. Chakrabarti, J. M. Kleinberg, and C. Faloutsos, “Realistic, mathematically tractable graph generation and evolution, using kronecker multiplication,” in *Proceedings of the 9th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD)*, Porto, Portugal, 2005, pp. 133–145.
- [14] W. Gray, J. Bogovic, J. Vogelstein, B. Landman, J. Prince, and R. Vogelstein, “Magnetic resonance connectome automated pipeline: An overview,” *Pulse, IEEE*, vol. 3, no. 2, pp. 42–48, 2012.
- [15] J. T. Vogelstein and C. E. Priebe, “Shuffled graph classification: Theory and connectome applications,” 2011.
- [16] S. V. N. Vishwanathan, N. N. Schraudolph, R. I. Kondor, and K. M. Borgwardt, “Graph kernels,” *JMLR*, vol. 11, pp. 1201–1242, 2010.
- [17] M. Fiedler, “Algebraic connectivity of graphs,” *Czechoslovak Mathematical Journal*, vol. 23, no. 98, pp. 298–305, 1973.
- [18] G. Li, M. Semerci, B. Yener, and M. Zaki, “Graph classification via topological and label attributes,” in *Proceedings of the 9th International Workshop on Mining and Learning with Graphs (MLG)*, San Diego, USA, Aug 2011.
- [19] O. Macindoe and W. Richards, “Graph comparison using fine structure analysis,” in *SocialCom/PASSAT*, 2010, pp. 193–200.
- [20] H. Elghawalby and E. R. Hancock, “Measuring graph similarity using spectral geometry,” in *Proceedings of the 5th international conference on Image Analysis and Recognition (ICIAR)*, 2008, pp. 517–526.
- [21] A. Kelmans, “Comparison of graphs by their number of spanning trees,” *Discrete Mathematics*, vol. 16, no. 3, pp. 241 – 261, 1976.
- [22] U. Kang, H. Tong, and J. Sun, “Fast random walk graph kernel,” in *Proceedings of the 12th SIAM International Conference on Data Mining (SDM)*, 2012.
- [23] S. Brin and L. Page, “The anatomy of a large-scale hypertextual web search engine,” *Computer Networks*, vol. 30, no. 1-7, pp. 107–117, 1998.
- [24] T. H. Haveliwalla, “Topic-sensitive pagerank: A context-sensitive ranking algorithm for web search.” *IEEE Transactions on Knowledge and Data Engineering*, vol. 15, no. 4, pp. 784–796, 2003.
- [25] P. G. Doyle and J. L. Snell, *Random Walks and Electric Networks*. Mathematical Association of America, 1984.
- [26] G. Jeh and J. Widom, “Simrank: a measure of structural-context similarity,” in *Proceedings of the 8th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, Edmonton, Alberta. ACM, 2002, pp. 538–543.
- [27] J. S. Yedidia, W. T. Freeman, and Y. Weiss, *Understanding belief propagation and its generalizations*. Morgan Kaufmann Publishers Inc., 2003, pp. 239–269.
- [28] D. Koutra, T.-Y. Ke, U. Kang, D. H. Chau, H.-K. K. Pao, and C. Faloutsos, “Unifying guilt-by-association approaches: Theorems and fast algorithms,” in *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD)*, Athens, Greece, 2011.
- [29] D. H. Chau, C. Nachenberg, J. Wilhelm, A. Wright, and C. Faloutsos, “Large scale graph mining and inference for malware detection,” in *Proceedings of the 11th SIAM International Conference on Data Mining (SDM)*, Mesa, AZ, 2011, pp. 131–142.
- [30] M. McGlohon, S. Bay, M. G. Anderle, D. M. Steier, and C. Faloutsos, “Snare: a link analytic system for graph labeling and risk detection,” in *Proceedings of the 15th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, Paris, France, 2009, pp. 1265–1274.
- [31] H.-N. Kim and A. El Saddik, “Personalized pagerank vectors for tag recommendations: inside folkrank,” in *Proceedings of the fifth ACM conference on Recommender systems*, 2011, pp. 45–52.

A Appendix

A.1 From Fast Belief Propagation (FABP) to DELTACON FABP ([28]) is a fast approximation of the loopy BP algorithm, which is guaranteed to converge and is described by the linear equation: $[\mathbf{I} + a\mathbf{D} - c'\mathbf{A}]\vec{s} = \vec{s}_0$,

where \vec{s}_0 is the vector of prior scores, \vec{s} is the vector of final scores (beliefs), $a = 4h_h^2/(1 - 4h_h^2)$, and $c' = 2h_h/(1 - 4h_h^2)$ are small constants, and h_h is a constant that encodes the influence between neighboring nodes. By (a) using the MacLaurin expansion for $1/(1 - 4h_h^2)$ and omitting the terms of power greater than 2, (b) setting $\vec{s}_0 = \vec{e}_i$ and $\vec{s} = \vec{s}_i$, and (c) setting $h_h = \epsilon/2$, we obtain eq. (2.1), the core formula of DELTACON.

A.2 Connection between FABP and Personalized RWR

THEOREM 2. *The FABP equation (2.1) can be written in the Personalized RWR-like form:*

$$[\mathbf{I} - (1 - c'')\mathbf{A}_*\mathbf{D}^{-1}]\vec{s}_i = c'' \vec{y},$$

where $c'' = 1 - \epsilon$, $\vec{y} = \mathbf{A}_*\mathbf{D}^{-1}\mathbf{A}^{-1}\frac{1}{\epsilon}\vec{e}_i$ and $\mathbf{A}_* = \mathbf{D}(\mathbf{I} + \epsilon^2\mathbf{D})^{-1}\mathbf{D}^{-1}\mathbf{A}\mathbf{D}$.

Proof. We begin from the derived FABP equation (2.1) and do simple linear algebra operations:

$$\begin{aligned} [\mathbf{I} + \epsilon^2\mathbf{D} - \epsilon\mathbf{A}]\vec{s}_i &= \vec{e}_i && (\times \mathbf{D}^{-1} \text{ from the left}) \\ [\mathbf{D}^{-1} + \epsilon^2\mathbf{I} - \epsilon\mathbf{D}^{-1}\mathbf{A}]\vec{s}_i &= \mathbf{D}^{-1}\vec{e}_i && (\mathbf{F} = \mathbf{D}^{-1} + \epsilon^2\mathbf{I}) \\ [\mathbf{F} - \epsilon\mathbf{D}^{-1}\mathbf{A}]\vec{s}_i &= \mathbf{D}^{-1}\vec{e}_i && (\times \mathbf{F}^{-1} \text{ from the left}) \\ [\mathbf{I} - \epsilon\mathbf{F}^{-1}\mathbf{D}^{-1}\mathbf{A}]\vec{s}_i &= \mathbf{F}^{-1}\mathbf{D}^{-1}\vec{e}_i && (\mathbf{A}_* = \mathbf{F}^{-1}\mathbf{D}^{-1}\mathbf{A}\mathbf{D}) \\ [\mathbf{I} - \epsilon\mathbf{A}_*\mathbf{D}^{-1}]\vec{s}_i &= (1 - \epsilon) (\mathbf{A}_*\mathbf{D}^{-1}\mathbf{A}^{-1}\frac{1}{1-\epsilon}\vec{e}_i) && \square \end{aligned}$$

A.3 Proofs for Section 3

LEMMA A.1. *The time complexity of DELTACON is linear on the number of edges in the graphs, i.e. $O(g \cdot \max\{m_1, m_2\})$.*

Proof. [Proof of Lemma 3.2] By using the Power Method [28], the complexity of solving eq. (2.1) is $O(|\mathcal{E}_i|)$ for each graph ($i = 1, 2$). The node partitioning needs $O(n)$ time; the affinity algorithm is run g times in each graph, and the similarity score is computed in $O(gn)$ time. Therefore, the complexity of DELTACON is $O((g+1)n + g(m_1 + m_2))$, where g is a small constant. Unless the graphs are trees, $|\mathcal{E}_i| < n$, so the complexity of the algorithm reduces to $O(g(m_1 + m_2))$. Assuming that the affinity algorithm is run on the graphs in parallel, since there is no dependency between the computations, DELTACON has complexity $O(g \cdot \max\{m_1, m_2\})$. \square

LEMMA A.2. *The affinity score of each node to a group (computed by DELTACON) is equal to the sum of the affinity scores of the node to each one of the nodes in the group individually (computed by DELTACON₀).*

Proof. Let $\mathbf{B} = \mathbf{I} + \epsilon^2\mathbf{D} - \epsilon\mathbf{A}$. Then DELTACON₀ consists of solving for every node $i \in \mathcal{V}$ the equation $\mathbf{B}\vec{s}_i = \vec{e}_i$; DELTACON solves the equation $\mathbf{B}\vec{s}_k = \vec{s}_{0k}$ for all groups $k \in (0, g]$, where $\vec{s}_{0k} = \sum_{i \in \text{group}_k} \vec{e}_i$. Because of the linearity of matrix additions, it holds true that $\vec{s}_k = \sum_{i \in \text{group}_k} \vec{s}_i$, for all groups k . \square

THEOREM 3. *DELTACON's similarity score between any two graphs G_1, G_2 upper bounds the actual DELTACON₀'s similarity score, i.e. $\text{sim}_{DC-0}(G_1, G_2) \leq \text{sim}_{DC}(G_1, G_2)$.*

Proof. [Proof of Theorem 1] Let $\mathbf{S}_1, \mathbf{S}_2$ be the $n \times n$ final-scores matrices of G_1 and G_2 by applying DELTACON₀, and $\mathbf{S}'_1, \mathbf{S}'_2$ be the respective $n \times g$ final-scores matrices by applying DELTACON. We want to show that DELTACON₀'s distance

$$d_{DC0} = \sqrt{\sum_{i=1}^n \sum_{j=1}^n (\sqrt{s_{1,ij}} - \sqrt{s_{2,ij}})^2}$$

is greater than DELTACON's distance

$$d_{DC} = \sqrt{\sum_{k=1}^g \sum_{i=1}^n (\sqrt{s'_{1,ik}} - \sqrt{s'_{2,ik}})^2}$$

or, equivalently, that $d_{DC_0}^2 > d_{DC}^2$. It is sufficient to show that for one group of DELTACON, the corresponding summands in d_{DC} are smaller than the summands in d_{DC_0} that are related to the nodes that belong to the group. By extracting the terms in the squared distances that refer to one group of DELTACON and its member nodes in DELTACON₀, and by applying Lemma A.2, we obtain the following terms:

$$t_{DC_0} = \sum_{i=1}^n \sum_{j \in \text{group}} (\sqrt{s_{1,ij}} - \sqrt{s_{2,ij}})^2$$

$$t_{DC} = \sum_{i=1}^n (\sqrt{\sum_{j \in \text{group}} s_{1,ij}} - \sqrt{\sum_{j \in \text{group}} s_{2,ij}})^2.$$

Next we concentrate again on a selection of summands (e.g. $i = 1$), we expand the squares and use the Cauchy-Schwartz inequality to show that

$$\sum_{j \in \text{group}} \sqrt{s_{1,ij} s_{2,ij}} < \sqrt{\sum_{j \in \text{group}} s_{1,ij} \sum_{j \in \text{group}} s_{2,ij}},$$

or equivalently that $t_{DC_0} > t_{DC}$. \square

A.4 Satisfaction of the Axioms Here we elaborate on the satisfiability of the axioms by DELTACON₀ and DELTACON.

A1. *Identity Property*: $\text{sim}(G_1, G_1) = 1$.

The proof is straightforward; the affinity scores are identical for both graphs.

A2. *Symmetric Property*: $\text{sim}(G_1, G_2) = \text{sim}(G_2, G_1)$.

The proof is straightforward for DELTACON₀. For the randomized algorithm, DELTACON, it can be shown that $\text{sim}(G_1, G_2) = \text{sim}(G_2, G_1)$ on average.

A3. *Zero Property*: $\text{sim}(G_1, G_2) \rightarrow 0$ for $n \rightarrow \infty$, where G_1 is the clique graph (K_n), and G_2 is the empty graph (i.e., the edge sets are complementary).

We restrict ourselves to a sketch of proof, since the proof is rather intricate.

Proof. [(Sketch of Proof - Zero Property)] First we show that all the nodes in a clique get final scores in $\{s_g, s_{ng}\}$, depending on whether they are included in group g or not. Then, it can be demonstrated that the scores have finite limits, and specifically $\{s_g, s_{ng}\} \rightarrow \{\frac{n}{2g} + 1, \frac{n}{2g}\}$ as $n \rightarrow \infty$ (for finite $\frac{n}{g}$). Given this condition, it can be directly derived that the ROOTED between the **S** matrices of the empty graph and the clique becomes arbitrarily large. So, $\text{sim}(G_1, G_2) \rightarrow 0$ for $n \rightarrow \infty$. \square

A.5 Satisfaction of the Properties Here we give some theoretical guarantees for the most important property, “edge importance” (P1). We prove the satisfiability of the property in a special case; generalizing this proof, as well as the proofs of the rest properties is theoretically interesting and remains future work.

Special Case [Barbell graph]: Assume A is a barbell graph with n_1 and n_2 nodes in each clique (e.g., B10 with $n_1 = n_2 = 5$ in Fig. 2), B does not have one edge in the first clique, and C does not have the “bridge” edge.

Proof. From eq. (2.1), by using the Power method we obtain the solution:

$$\vec{s}_i = [\mathbf{I} + (\epsilon \mathbf{A} - \epsilon^2 \mathbf{D}) + (\epsilon \mathbf{A} - \epsilon^2 \mathbf{D})^2 + \dots] \vec{e}_i \Rightarrow$$

$$\vec{s}_i \approx [\mathbf{I} + \epsilon \mathbf{A} + \epsilon^2 (\mathbf{A}^2 - \mathbf{D})] \vec{e}_i,$$

where we ignore the terms of greater than second power. By writing out the elements of the $\mathbf{S}_A, \mathbf{S}_B, \mathbf{S}_C$ matrices as computed by DELTACON₀ and the above formula, and also the ROOTED between graphs A, B and A, C, we obtain the following formula for their relevant difference:

$$d(A, C)^2 - d(A, B)^2 = 2\epsilon \{ \epsilon(n - f) + 1 - (\frac{2\epsilon^3(n_1 - 2)}{c_1^2} + \frac{\epsilon}{c_2^2}) \},$$

where $c_1 = \sqrt{\epsilon + \epsilon^2(n_1 - 3)} + \sqrt{\epsilon + \epsilon^2(n_1 - 2)}$ and $c_2 = \sqrt{\epsilon^2(n_1 - 2)} + \sqrt{\epsilon + \epsilon^2(n_1 - 2)}$. We can show that $c_1 \geq 2\sqrt{\epsilon}$ for $h_h < 1$ (which holds always) and $c_2 \geq \sqrt{\epsilon}$, where $f = 3$ if the missing edge in graph B is adjacent to the “bridge” node, and $f = 2$ in any other case. So, $d(A, C)^2 - d(A, B)^2 \geq 0$.

This property is not always satisfied by the euclidean distance. □